VOXSPELL

A Spelling Aid with Real Time Feedback

Emily Yee Vern Chan
SOFTENG 206
University of Auckland

***Executive Summary*—VOXSPELL is a spelling aid tool that provides real-time feedback to support its users while they practice spelling words to improve their vocabulary. It features audio-visual learning, a configurable speech-synthesiser, a scoring system to measure the user's learning progress and the ability to customize the list of words to spell. This application is designed for second-language learners, aged between 18 to 25 years, who intend to pick up English as their second language. The report first presents a discussion regarding the design considerations and justifications for the application's graphical user interface, followed by a similar discussion but for its functionality and features instead. Then, an overview of the application's code design and development process are presented, followed by an overview of the testing process and peer evaluations. Finally, potential future work to improve the application and conclusions are presented.**

I. Introduction

The VOXSPELL project was aimed to produce a spelling aid tool that provides real-time feedback to support its users while they practice spelling words to improve their vocabulary. The application's target user demographic is second-language learners, aged between 18-25 years, who intend to pick up English as their second language.

Simplistic and utilitarian visual designs suit the target users as they are only intent to use the application for the sole purpose of improving their spelling. This was achieved by organising the display layout such that it takes advantage of the natural gaze of the user. This was also achieved by showing only one out of five possible screens at any given time. Visual contrast and large fonts not only achieves the desired visual design as well, but also accommodates for visually impaired users.

The application's functionality and features were designed to aptly suit the client and target users' requirements. Key functionality and features are highlighted below:

- categorising quiz words as parts of speech in the English language
- spaced repetition algorithm to support the learning process efficiently
- scoring mechanism to quantifiably measure the user's progress
- speech synthesiser playback and accent adjustment
- slideshow video consisting of words of the recent quiz game
- changing the list of words to be quizzed on

To develop this application, the Java programming language, in conjunction with other libraries such as VLCJ, Swing/AWT, Festival and FFMPEG, were used. The development process was supported by software tools such as WindowBuilder, Git and GitHub. The application is limited for use in Linux operating systems only as the Festival and FFMPEG libraries are only available in those systems.

Due to time constraints, some faults still remain in the application and some usability features were not implemented. However, they were deemed unimportant at the present time as the application already aptly satisfies more than just the basic requirements. Future improvements to the application include more extensive testing to reduce faults and implementing the aforementioned features.

II. Visual Design

A. Choice of programming language and packages used

The programming language employed in this project was Java due to its global widespread usage and compatibility with various operating systems.

The application is suited for Linux operating systems only. This is because Linux has the BASH functionality, which is necessary for using the Festival system. The Festival system is required by the application for speech synthesis during a spelling quiz.

Java provides a widget toolkit to support the creation of the application's graphical user interface (GUI). The toolkit includes the Swing and AWT packages. These packages were used for the following two reasons:

1. Customizable appearance design

The Swing package is platform-independent. Therefore, the application's appearance can be customized instead of using the standard native OS GUI components.

2. Model/View/Controller software design pattern

Both packages promote the use of the Model/View/Controller software design pattern, which means the data being seen from the user interface controls is decoupled from the controls themselves.

B. *Colour consideration, display layout and font*

The overall visual appearance that suits the targeted user demographic should be simplistic and utilitarian. This is because the users are only intent to use the application for the sole purpose of improving their spelling.

An apt exemplar would be Google's front page, which is also simplistic, utilitarian and provides the sole purpose of web page searching. Google applies highly saturated colours at key-focus areas and a plain white colour as the background. Hence, this best practice is applied for the visual design of this application.

1. Colour consideration

Visual contrast is necessary for better visibility when using the application, especially for visually impaired users. To achieve visual contrast, exaggerated colour light differences and highly saturated colours were used.

A neutral-white colour was used for the background for all screens. This is so that the user's visual focus is attracted to the areas with colour and to diminish eye fatigue.

The highly saturated colours were used consistently in the application. Each functionality has its own designated colour: green for 'New Spelling Quiz', blue for 'Scoreboard', yellow for 'Change Word List' and maroon for 'Help'. As each functionality has its own screens, its corresponding colour is used consistently throughout the screen. For example, all buttons in the 'New Spelling Quiz' screen has the same green border as the 'New Spelling Quiz' button in the main screen.

2. Display layout

Users typically read in the order of left to right, then top to bottom of a screen [1]. This theory is applied in this application where the components are arranged in the same order to emphasise its priority.

An example would be when the user wants to change some words in the word list. The user first needs to choose a category to change its words. Hence, the functionality to do this is placed near the top of the screen. Then the user needs to provide the words to add and remove from the category, which is the main purpose of the screen. Hence, the functionality to achieve this is placed right in the centre of the screen (below the 'choose category functionality') and takes up the most space.

3. Font

The font is also used consistently throughout the application. Its size is slightly larger than the typical size 12 that appears in most texts for users aged 18-24. This is intentional for the ease-of-use for visually impaired users.

C. *Presentation of information*

Swing/AWT's CardLayout is used as the layout manager for the main program. This layout enables the interchangeability of screens in the application. Interchanging the screens allows the user to be easily guided through using each main functionality in the application. It also allows for the user to focus on using one main functionality at a time. Thus, this layout prevents overloading the user with functions and features all at once.

Each screen is only presented when the user intends to use its functionality. For example, the scoreboard screen is presented only when the user wants to see the latest scores.

D. *Other interface issues*

While the CardLayout is used as the main program's layout manager, Swing/AWT's AbsoluteLayout is used to customise the appearance of each screen. The screens' GUI appears satisfactorily when using the application in 16:9 display ratio computers, but not in any other ratios.

It should be noted that most computers use the 16:9 ratio (including 14" notebooks, 15.6" laptops, 18.5" monitors, 21.5" monitors, 23" monitors, and 1080p televisions) [2]. Future improvements could include enabling the application to suit other display ratios.

III. FUNCTIONALITY

This section details how the client's functionality requirements were achieved. It also details the additional functionality features included in the application that was complementary to the basic functionality. All additional features were discussed with the client and improve the application's usability.

A. *New Spelling Quiz*

1. Categories of words

The spelling list is categorised into four parts of speech of a language: adjectives, adverbs, nouns and verbs. Other parts of speech, such as articles (e.g. a, and, the etc.) and pronouns (e.g. I, me, you, she, he etc.), are excluded as they do not have an extensive number of possible words. The intention behind this categorisation is to support the target users' learning of a second language.

2. Spaced Repetition Algorithm

It has been proven that spaced repetition improves memory retention [3]. Hence, a spaced repetition algorithm was applied with the aim to quiz the user on more unfamiliar words than familiar words. This is to help the target users achieve their goal of picking up a new language efficiently as well as compiling the quiz's words according to the user's current ability.

This algorithm also provides an indication when the user has become familiar with most of the words in the category. A notification is given to the user to add more words to the category (via the 'Change Word List' functionality) to further expand the user's vocabulary.

3. Scoring mechanism

The scoring mechanism involves recording the number of times the user has 'mastered', 'faulted' or 'failed' in spelling the quiz word. These records will be used for the scoreboard (see Scoreboard section). The mechanism also involves keeping track of how familiar the user is with the quiz word by assigning it a 'level of familiarity'. The lowest level of familiarity is 1 and the highest is 5. The level of familiarity is only recorded for purposes of the spaced repetition algorithm and not for the scoreboard.

When the user has spelled the word correctly on the first attempt, the user has mastered the word and the word increases by one level of familiarity.

When the user has spelled the word wrongly on the first attempt, the user has faulted the word. The level of familiarity remains the same.

When the user has spelled the word wrongly on the second attempt, the user has failed the word and the word decreases by one level of familiarity. Also, the correct spelling of the word is shown to the user.

4. Error handling

The application responds appropriately when an error occurs. During the quiz, the user may mistakenly give a blank input or non-alphabetic characters (other than one apostrophe) when spelling a word. The application simply informs the user of the mistake and does not account it as a faulted or failed word. At most one apostrophe is allowed when the word actually contains an apostrophe (e.g. I'll, you'll, won't). If the quiz word does not contain an apostrophe, an input with any number of apostrophes input is considered invalid.

5. Visual overview of current quiz progress

The user can see a visual overview of the current quiz progress, showing how many words the user has left to spell and how many words have been spelt correctly/wrongly so far.

For example, when the user has mastered the third word in the quiz, the third black circle shape from the left turns into a green check mark. If the user had faulted the word, it turns into a red cross mark instead. Using both shapes and colours together helps to improve usability and visibility, as even colour-blind users can also see the visuals as well.

6. Sound effects

Sound effects are played when the user has spelt the word correctly/wrongly to enhance the application's appeal.

7. Relisten to word

The user may listen to the word for any number of times without penalty. This accommodates for circumstances where the user did not catch the word properly, thereby improving the application's usability.

8. Changing the speech synthesiser's accent

The user may change the speech synthesiser's accent (current choices available are American and New Zealand accents). This functionality is included to achieve two purposes:

- The user may get bored of hearing the same accent, so this functionality increases the application's appeal.
- Some words are pronounced differently with different accents. For example, the word 'tomato' can be pronounced 'to-mah-to' or 'to-may-to'. Hence, hearing the word in different accents would be pragmatic.

9. Video reward at the end of each quiz

It has been proven that audio visual learning tools enables users to learn more effectively [4]. Hence, this theory is applied in this application. After the user has finished spelling 10 words, the user may choose to play a video reward. The video reward consists of a slideshow with background music showing all the words in the recent quiz.

B. Scoreboard

The scoreboard provides a high-level view and a low-level view of the scores for each category. The purpose of the scoreboard is to provide a quantifiable measurement of the user's learning. The scoreboard also serves as a motivation for the user to continue spelling more words, and thereby continue using the application.

The high-level view is shown through a progress bar, indicating the progress of the user towards achieving 100% accuracy for every word in the category. The low-level view is shown through a table. The table lists all of the words in the category. For each word, the table includes its corresponding number of times mastered/faulted/failed and its accuracy percentage. For ease of use, the table is sorted by decreasing accuracy percentage, then by increasing alphabetical order.

C. Changing Word List

1. Two possible ways to change the word list

Spelling the same list of words repeatedly would make the user lose interest in the application. Furthermore, it is in the target users' best interest to expand their vocabulary and be challenged at a suitable difficulty according to their current ability. Hence, the application includes a functionality for changing the word list in two ways:

- Changing the entire word list: The user may provide his/her own custom list of words.
- Changing some words in the current word list: The user may add and/or remove some words from the current word list.

2. Error handling

For both functionalities, the application responds appropriately when an error occurs. The user errors that the application accounts for are as follows:

- There are more than one apostrophes in a word
- The word contains at least one whitespace character
- The word contains a non-alphabetical character (other than one apostrophe)
- The new word list file contains a blank line.

- The new word list file has more/less categories specified than necessary.
- The word that the user wants to add to the current word list already exists.
- The word that the user wants to remove from the current word list doesn't exist.

Both functionalities also have a high degree of error tolerance in preventing the user from executing irreversible actions:

- When the user is changing the entire word list, a warning message is shown to inform the user that changing the entire word list will remove all previous words, scores and progress in memory.
- When the user is changing only some parts of the word list and unintentionally clicked 'Cancel Changes', a confirmation message is shown to the user.

D. Additional usability design decisions

With reference to the guidance in regards to improving usability given in the article "What Does Usability Mean: Looking Beyond 'Ease of Use'" [5], each functionality shown in the application is intuitive and self-explanatory. If the user still needs more help, they can hover the mouse cursor over the functionality and view its tool tip text. Furthermore, the user may refer to the 'Help' screen which can be accessed from the main menu. The user manual is also highly comprehensive to aid the user.

IV. Code design and development

A. Documentation of software design

The Java programming language was the best choice for the project, given the scope and developers' abilities. The language incorporates software design features such as object orientation, interfaces, abstract classes and anonymous inner classes. These features enabled software best practices, such as design patterns, to be applied to the project. Furthermore, due to Java's cross-platform compatibility, the Festival speech synthesiser and the FFMPEG software project (which are only available in Linux operating systems) could be utilised.

The following libraries were used in conjunction with the application:

- VLCJ
- Swing/AWT
- Festival
- FFMPEG

B. Development process

The development process was structured as per the University of Auckland's SOFTENG 206 course programme. My colleague, Chen Li, and I developed the prototype of the application, which includes implementations for the basic layout and core functionality. However, I had single-handedly developed the beta version of the application, which was peer reviewed by four of my fellow colleagues.

C. Software tools

Several software tools were utilised to support the several aspects of the development process:

1. Tool used: WindowBuilder
   Aspect of process supported: UI Modelling

While developing the beta version of the application, the GUI layout and design was first modelled using Eclipse's WindowBuilder tool. After the model was satisfactory, its functionalities were implemented.

2. Tool used: Git and GitHub
   Aspect of process supported: Version Control

While developing the application prototype, Git and GitHub were used to manage the project's version control. This enabled smoother simultaneous collaboration with my colleague. Even during the solo development of the beta version, these tools were still used as they also enabled easier management of the project's workflow.

V. Testing and Peer Evaluation

A. Testing

After each incremental functionality or feature was implemented, the application was tested to ensure that the incremental functionality/feature produced the expected basic behaviour without failure. Then the application was tested as a whole with the incremental functionality/feature, to ensure smooth integration.

As VOXSPELL is an application with complex functionality, features and GUI, it was infeasible to extensively test the application. However, sufficient testing was completed to ensure that the application's basic functionality has no faults.

B. Peer Evaluation

The beta version of the application was peer-evaluated by four of my colleagues. Their evaluations presented highly insightful feedback as well as revealed faults that I had overlooked.

Each fault from the evaluations was either resolved or unresolved for justified reasons (detailed below).

1. The following lists the faults discovered by my colleagues, all of which have been resolved:
   a. The speech synthesiser in the New Zealand accent speaks too quickly, even though the American accent speaks at the correct pace.
   b. Unable to correctly edit the entered entries when trying to change the current word list.
   c. Scoreboard sorts accuracy percentages as strings instead of integers (i.e. 50% comes before 100%, because 50% has less characters).

   d. The visual overview progress in the quiz screen does not reset when the user starts a second quiz in the same session.

   e. The warning message is not emphasised enough when the user attempts to change the entire word list.

2. The following lists the faults which were unresolved, with their respective justified reasons:

   a. Lack of appealing transition effects between slides in slideshow video: due to time constraints, this feature was not implemented. Implementing this feature would mean creating a short video for each slide and then link all of those videos in order. However, this is listed as an item under the 'Future Work' section.

   b. Allow application window to be resizable: again, due to time constraints, this feature was not implemented as it would mean having to refactor the layout manager (which is currently AbsoluteLayout) for all screens to a GridBagLayout. Once again, this is listed as an item under the 'Future Work' section.

   c. When the user wants to change the entire word list, the user could potentially specify the categories in the wrong order. This leads to the application adding the words to the wrong category. At present, a solution to resolve this fault is not found. However, a workaround solution is provided to the user. The solution includes comprehensively detailing and emphasising how the categories should be specified in the file in the user manual and help screen. Furthermore, a template input file was provided in the package together with the application.

VI. Future Work

During the development process, my colleagues and I had other full-time commitments and obligations. Hence, limited time was available to conduct the project. As such, certain tasks were prioritised over others. Consequently, some faults still remain in the application program and some features were not implemented. However, the aforementioned faults were either deemed not critical or a workaround solution was provided. The features were deemed as ideal, but not necessary for the application. These features include:

- Having appealing transition effects between slides in the video slideshow
- Allowing the application to be resizable and accommodate for computer display ratios other than 16:9.
- Having a more robust error handling for the situation where the user may list the order of categories incorrectly when changing the entire word list.

The VOXSPELL application currently satisfies the basic expectations of the client and target users, with additional functionality and features. Hence, future developmental work would be largely directed at reducing faults through additional testing. Also, the aforementioned features could be implemented in a future release to further improve the application's usability and functionality.

VII. Conclusion

Simplistic and utilitarian visual designs suit the target users as they are only intent to use the application for the sole purpose of improving their spelling. This was achieved by organising the display layout such that it follows the natural gaze of the user as well as showing only one out of five possible screens at any given time. Visual contrast and large fonts not only achieves the desired visual design as well, but also accommodates for visually impaired users.

The application's functionality and features were designed to aptly suit the client and target users' requirements. Key functionality and features are highlighted in the following list:

- categorising quiz words as parts of speech in the English language
- spaced repetition algorithm to support the learning process efficiently
- a scoring mechanism to quantifiably measure the user's progress
- speech synthesiser playback and accent adjustment
- slideshow video consisting of words of the recent quiz game
- changing the list of words to be quizzed on

The programming language used to develop this application was Java. The additional libraries used were VLCJ, Swing/AWT, Festival and FFMPEG. The tools used to support the development process were WindowBuilder, Git and GitHub. As Festival and FFMPEG libraries are only available in Linux operating systems, the application may only be used in that system.

Due to time constraints, some faults still remain in the application and some features were not implemented. However, they were deemed unimportant. Future improvements to the application include more extensive testing to reduce faults and implementing the aforementioned features to improve the application's usability and functionality.

Credits and Acknowledgement

This application credits the following royalty-free sources for its media files:

1. Main menu title and logo

Square graphic by Freepik from Flaticon is licensed under CC BY 3.0. Made with LogoMaker.

2. New spelling quiz sound effects:

http://www.orangefreesounds.com/correct-answer-sound-effect/

https://www.freesound.org/people/Bertrof/sounds/131657/

3. End of video background music:

http://www.purple-planet.com

All other media files in the application are originals.

REFERENCES

[1] C. O'Connell, "Eyetracking and Web site Design", *Usability.gov*, 2010. [Online]. Available: https://www.usability.gov/get-involved/blog/2010/03/eyetracking.html. [Accessed: 24- Oct- 2016].

[2] "Screen resolution statistics", *Rapidtables.com*, 2014. [Online]. Available: http://www.rapidtables.com/web/dev/screen-resolution-statistics.htm. [Accessed: 24- Oct- 2016].

[3] D. Ausubel and M. Youssef, "The Effect of Spaced Repetition on Meaningful Retention", *The Journal of General Psychology*, vol. 73, no. 1, pp. 147-150, 1965.

[4] S. Rasul, Q. Bukhsh and S. Batool, "A study to analyze the effectiveness of audio visual aids in teaching learning process at uvniversity level", *Procedia - Social and Behavioral Sciences*, vol. 28, pp. 78-81, 2011.

[5] W. Quesenbery, "What Does Usability Mean: Looking Beyond 'Ease of Use' - Whitney Interactive Design", *Wqusability.com*, 2011. [Online]. Available: http://www.wqusability.com/articles/more-than-ease-of-use.html. [Accessed: 24- Oct- 2016].